

WHAT IS CLAIMED IS:

- 1 1. A method comprising:
 - 2 requesting access to a shared resource for a first process having a first
 - 3 local priority;
 - 4 determining if a second process is simultaneously requesting access to
 - 5 the shared resource, the second process having a second local
 - 6 priority; and
 - 7 if the second process is simultaneously requesting access to the shared
 - 8 resource, then granting access one of the first priority and the
 - 9 second priority having a higher local priority.
- 1 2. The method of claim 1, wherein the local priority is fixed for each of the
- 2 first and the second process.
- 1 3. The method of claim 1, additionally comprising if the second process is not
- 2 simultaneously requesting access to the shared resource, then:
 - 3 determining if the second process currently has a lock on the shared
 - 4 resource;
 - 5 if the second process currently has a lock on the shared resource, then
 - 6 denying the first process access to the shared resource; and
 - 7 if the second process does not have a lock on the shared resource, then
 - 8 granting the first process access to the shared resource.

1 4. A method comprising:
2 requesting access to a shared resource for a first process having a first
3 local priority, and a first wait time;
4 determining if a second process is simultaneously requesting access to
5 the shared resource, the second process having a second local
6 priority, and a second wait time;
7 if the second process is simultaneously requesting access to the shared
8 resource, then granting access to one of the first process and the
9 second process having a longer wait time; and
10 if the first wait time equals the second wait time, then granting access to
11 one of the first process and the second process having a local
12 higher priority.

1 5. The method of claim 4, wherein the local priority is fixed for each of the
2 first and the second process.

1 6. The method of claim 4, additionally comprising if the second process is not
2 simultaneously requesting access to the shared resource, then:
3 determining if the second process currently has a lock on the shared
4 resource;
5 if the second process currently has a lock on the shared resource, then
6 denying the first process access to the shared resource; and

7 if the second process does not have a lock on the shared resource, then
8 granting the first process access to the shared resource.

1 7. A method comprising:

2 requesting access to a shared resource for a first process having a first
3 global priority on a global priority queue of a global arbiter;

4 determining if a second process is simultaneously requesting access to
5 the shared resource, the second process having a second global
6 priority on the global priority queue of the global arbiter; and

7 if the second process is simultaneously requesting access to the shared
8 resource, then granting access to one of the first process and the
9 second process having a higher global priority.

1 8. The method of claim 7, wherein the global priority queue is one of a
2 plurality of global priority queues in the global arbiter, and each global
3 priority queue corresponds to a given shared resource.

1 9. The method of claim 7, additionally comprising if the second process is not
2 simultaneously requesting access to the shared resource, then:

3 determining if the second process currently has a lock on the shared
4 resource;

5 if the second process currently has a lock on the shared resource, then
6 denying the first process access to the shared resource; and

7 if the second process does not have a lock on the shared resource, then
8 granting the first process access to the shared resource.

1 10. A method comprising:

2 requesting access to a shared resource for a first process having a first
3 global priority on a global priority queue of a global arbiter, and
4 having a first wait time;

5 determining if a second process is simultaneously requesting access to
6 the shared resource, the second process having a second global
7 priority on the global priority queue of the global arbiter, and having
8 a second wait time;

9 if the second process is simultaneously requesting access to the shared
10 resource, then granting access to one of the first process and the
11 second process having a longer wait time; and

12 if the first wait time is equal to the second wait time, then granting access
13 to one of the first process and the second process having a higher
14 than global priority.

1 11. The method of claim 10, wherein the global priority queue is one of a
2 plurality of global priority queues in the global arbiter, and each global
3 priority queue corresponds to a given shared resource.

1 12. The method of claim 10, additionally comprising if the second process is
2 not simultaneously requesting access to the shared resource, then:

3 determining if the second process currently has a lock on the shared
4 resource;
5 if the second process currently has a lock on the shared resource, then
6 denying the first process access to the shared resource; and
7 if the second process does not have a lock on the shared resource, then
8 granting the first process access to the shared resource.

1 13. An apparatus comprising:
2 a local arbiter to arbitrate on behalf of the corresponding process for one
3 of a plurality of resources; and
4 a semaphore to indicate a status of the corresponding process.

1 14. The apparatus of claim 13, additionally comprising a local priority block to
2 indicate a local priority of the corresponding process.

1 15. The apparatus of claim 13, additionally comprising a timer element to
2 determine a wait time for the corresponding process.

1 16. A system comprising:
2 one or more shared resources; and
3 one or more processes, each corresponding to a semaphore system, and
4 each semaphore system having a local arbiter to arbitrate for
5 access to a given one of the shared resources.

1 17. The system as in claim 16, wherein a given semaphore system
 2 additionally comprises a local arbiter block having a local priority
 3 corresponding to a corresponding process, and the local arbiter arbitrates
 4 for access to a given one of the shared resources by granting access to
 5 the corresponding process if its corresponding process has a local global
 6 priority than a conflicting process.

1 18. The system of claim 17, wherein the local priority is fixed.

1 19. The system as in claim 17, wherein the given semaphore system
 2 additionally comprises a timer element, and the local arbiter arbitrates for
 3 access to a given one of the shared resource by:
 4 granting access to the corresponding process if the corresponding
 5 process waited longer for the given resource than the conflicting
 6 process; and
 7 if the corresponding process waited the same amount of time for the given
 8 resource as the conflicting process, then granting access to the
 9 corresponding process if the corresponding process has a higher
 10 local priority than the conflicting process.

1 20. The system of claim 16, wherein the local arbiter arbitrates for access to a
 2 given one of the shared resources by granting access to the
 3 corresponding process if there are no conflicting processes.

1 21. The system as in claim 20, the system additionally comprising a global
2 arbiter having a global priority queue, the global arbiter to:
3 modify process priorities by moving processes that have been granted
4 access to a given resource to a position in the global priority queue
5 having a lowest priority; and
6 arbitrate conflicts between a first process and a second process by
7 granting access to one of the first process and the second process
8 having a having a higher global priority.

1 22. The system of claim 16, wherein the semaphore additionally comprises a
2 timer element, and the local arbiter arbitrates for access to a given one of
3 the shared resources by:
4 granting access to the corresponding process if the corresponding
5 process waited longer for the given resource than a conflicting
6 process; and
7 if the corresponding process has waited the same amount of time for the
8 given resource as the conflicting process, then offloading the
9 arbitration process to a global arbiter.

1 23. The system as in claim 22, the system additionally comprising the global
2 arbiter having a global priority queue, the global arbiter to:
3 modify priorities to processes by moving processes that have been
4 granted access to a given resource to a position in the global

5 priority queue having a lowest priority; and
6 arbitrate conflicts between a first process and a second process by
7 granting access to one of the first process and the second process
8 having a higher priority.

1 24. A machine-readable medium having stored thereon data representing
2 sequences of instructions, the sequences of instructions which, when
3 executed by a processor, cause the processor to perform the following:
4 request access to a shared resource for a first process having a first local
5 priority;
6 determine if a second process is simultaneously requesting access to the
7 shared resource, the second process having a second local priority;
8 and
9 if the second process simultaneously requests access to the shared
10 resource, then grant access one of the first priority and the second
11 priority having a higher local priority.

1 25. The machine-readable medium of claim 24, wherein the local priority is
2 fixed for each of the first and the second process.

1 26. The machine-readable medium of claim 24, additionally comprising if the
2 second process is not simultaneously requesting access to the shared
3 resource, then additionally comprising sequences of instructions which,
4 when executed by a processor, cause the processor to perform:

5 determine if the second process currently has a lock on the shared
6 resource;
7 if the second process currently has a lock on the shared resource, then
8 deny the first process access to the shared resource; and
9 if the second process does not have a lock on the shared resource, then
10 grant the first process access to the shared resource.

1 27. An apparatus comprising:
2 at least one processor; and
3 a machine-readable medium having instructions encoded thereon, which
4 when executed by the processor, are capable of directing the
5 processor to:
6 request access to a shared resource for a first process having a first local
7 priority;
8 determine if a second process is simultaneously requesting access to the
9 shared resource, the second process having a second local priority;
10 and
11 if the second process simultaneously requests access to the shared
12 resource, then grant access one of the first priority and the second
13 priority having a higher local priority.

1 28. The apparatus of claim 27, wherein the local priority is fixed for each of the
2 first and the second process.

1 29. The apparatus of claim 27, additionally comprising if the second process is
2 not simultaneously requesting access to the shared resource, then
3 additionally encoded instructions which, when executed by a processor,
4 are capable of causing the processor to:

5 determine if the second process currently has a lock on the shared
6 resource;

7 if the second process currently has a lock on the shared resource, then
8 deny the first process access to the shared resource; and

9 if the second process does not have a lock on the shared resource, then
10 grant the first process access to the shared resource.

1 30. An apparatus comprising:

2 means for arbitrating on behalf of the corresponding process for one of a
3 plurality of resources; and

4 means for indicating a status of the corresponding process.

1 31. The apparatus of claim 30, additionally comprising means for indicating a
2 local priority of the corresponding process.

1 32. The apparatus of claim 30, additionally comprising means for determining
2 a wait time for the corresponding process.